Applicant : Gilbert Wolrich et al.
Serial No. : 10/070,908
Filed : July 3, 2002
Page : 6 of 12

Attorney's Docket No : 10559-311US1 / P9632US
Intel Corporation

REMARKS

Applicant amended claim 1 to add features similar to the features previously recited in claims 2 and 14.

Claim 1 now calls for "…to prefetch, prior to performing an evaluation of a branch condition associated with the branch instruction, an instruction for the "branch taken" condition rather than a next sequential instruction." ,e.g., a "branch guess" operation that is performed if the first token specifies a zero or one instructions to execute after the branch, and to remove the recitation "a branch based on a bit of register being set or cleared, the bit and register being specified in [the branch instruction]". Support for claim amendments is provided, for example, at page 10, line 24, to page 12, line 15 of the application.

Applicant similarly amended independent claims 17 and 22, and cancelled claims 2, 14, 18 and 23. Additionally, applicant amended claims 6 and 19-21 for greater clarity and to make the language recited in those claims consistent with the amended language of the independent claims. After these amendments, claims 1, 3-4, 6-8, 10-13, 17, 19-22 and 24-26 are pending in the above-identified application. Claims 1, 17, and 22 are independent.

The examiner rejected claims 1-4, 5, 6-9, 11-14 and 17-23 under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,928,358 to Takayama, in view of U.S. Patent No. 5,724,563 to Hasegawa. The examiner also rejected claim 10 under 35 U.S.C. §103(a) as being unpatentable over Takayama in view of Hasegawa, and further in view of U.S. Patent No. 5,274,770 to Khim Yeoh et al. The examiner further rejected claims 24-26 under 35 U.S.C. §103(a) as being unpatentable over Takayama in view of Hasegawa, and further in view of U.S. Patent No. 5,923,872 to Chrysos.

Applicant's amended independent claim 1 recites "… executing a branch instruction … including a first token that specifies the number of instructions in the instruction stream … after the branch instruction to execute before performing the branch operation and a second token that specifies a branch guess operation to cause the processor, if the first token specifies zero or one instructions to execute after the branch, to prefetch, prior to performing an evaluation of a branch condition associated with the branch instruction, an instruction for the "branch taken" condition rather than a next sequential instruction."

Thus, performance of the branch guess operation is dependent on the value of the specified value corresponding to the instruction's branch defer option. As explained in applicant's Specification:

> The microengine can combine branch guessing with 1-cycle branch deferment to improve the result further. For guess branch taken with 1-cycle deferred branch/branch is taken is in Table 10.
>
> **Table 10**
>
> ```
>                  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
> -----------------+---+---+---+---+---+---+---+---+
> microstore lookup |n1 |cb |n2 |b1 |b2 |b3 |b4 |b5 |
> reg addr gen      |   |n1 |cb |n2 |b1 |b2 |b3 |b4 |
> reg file lookup   |   |   |n1 |cb |n2 |b1 |b2 |b3 |
> ALU/shifter/cc    |   |   |   |n1 |cb |n2 |b1 |b2 |
> write back        |   |   |   |   |n1 |cb |n2 |b1 |
> ```
>
> In the case above, the 2 cycles of branch latency are hidden by the execution of n2, and by correctly guessing the branch direction.
>   If microcode guesses incorrectly, 1 cycle of branch latency remains exposed as in Table 11 (guess branch taken with 1-cycle deferred branch/branch is NOT taken). (Page 10, line 24, to page 11, line 16 of the application)

Thus, the described microengine enables hiding at least one cycle of branch latency (two if a correct branch guess was made). This advantage is achieved in situations where the branch guess option is used together with a branch defer that is set to 1 or 0.

In contrast, none of the prior art references cited by the examiner discloses or suggest at least the feature of "executing a branch instruction in execution of an instruction stream with the branch instruction including a first token that specifies the number of instructions in the instruction stream that are after the branch instruction to execute before performing the branch operation and a second token that specifies a branch guess operation to cause the processor, if the first token specifies zero or one instructions to execute after the branch, to prefetch, prior to performing an evaluation of a branch condition associated with the branch instruction, an instruction for the "branch taken" condition rather than a next sequential instruction."

Specifically, Takayama describes an apparatus to facilitate branching prediction functionality. With respect to the branch instruction illustrated in FIG. 3, Takayama explains that:

> FIG. 3 shows the branch instruction format of the information processing apparatus 100. A branch instruction 20 is composed of a 13-bit operation code 20a, 1-bit branch prediction information 20b, 2-bit branch history information 20c and a 16-bit branch destination address 20d. The operation

> code 20a shows an operation code and a branch condition which identify
> the present instruction.
>
> The branch prediction information 20b predicts whether the branch is
> taken or not taken when the present branch instruction is next executed.
> The relation between the branch prediction information 20b and the
> prediction is as follows: when the branch prediction information 20b is "0",
> a branch is predicted "not taken"; and when the branch prediction
> information 20b is "1", a branch is predicted "taken". (Col. 7, lines 38-44)

While Takayama's branch instruction specifies branch prediction functionality, at no point does Takayama describe that such a branch prediction functionality operates in concert with a branch deferred functionality. Takayama certainly does not describe a branch instruction to perform a branch prediction (or branch guess) that is performed based on the value of a branch deferred functionality being set to one or zero instructions to execute after the branch instruction. Accordingly, Takayama fails to describe and suggest at least the feature of "executing a branch instruction in execution of an instruction stream with the branch instruction including a first token that specifies the number of instructions in the instruction stream that are after the branch instruction to execute before performing the branch operation and a second token that specifies a branch guess operation to cause the processor, if the first token specifies zero or one instructions to execute after the branch, to prefetch, prior to performing an evaluation of a branch condition associated with the branch instruction, an instruction for the "branch taken" condition rather than a next sequential instruction."," as required by applicant's independent claim 1.

Hasegawa discloses a pipeline processor that can execute predictive branch instructions (Abstract). Hasegawa refers to its branch instruction as "predictive branch instructions", and explains that:

> FIG. 2 shows a format for the branch instruction used in the pipeline
> processor 100. In this specification, the branch instruction shown in FIG. 2
> is called a "predictive branch instruction". A "predictive branch
> instruction" is an instruction to branch to a branch target address after a
> predetermined number of instructions are executed from the "predictive
> branch instruction". The predetermined number is given as an operand of
> the predictive branch instruction. (emphasis added, col. 5, lines 54-61)

Thus, Hasegawa's predictive branch instructions are branch instructions that perform branches after executing a specified number of instruction (i.e., Hasegawa's branch instruction is a delayed branch). Hasegawa further describes that the format of its predictive branch instructions includes a region 21 that stores an opcode, a region 22 for specifying a branch target

Applicant : Gilbert Wolrich et al.   Attorney's Docket No.: 10559-311US1 / P9632US
Serial No. : 10/079,008   Intel Corporation
Filed   : July 3, 2002
Page   : 9 of 12

address, and a region 23 for storing the number of at least one instruction which is to be executed in succession after the predictive branch instruction and before the control flow is changed (FIG. 2 and col. 6, lines 1-6).

At no point, however, does Hasegawa describe a branch guess operation that is performed to cause a prefetch of an instruction corresponding to the branch taken condition associated with the branch instruction. Moreover, Hasegawa certainly does not describe that such a branch guess operation is performed in circumstances where the branch instruction specifies that one or zero instruction are to be performed after the branch instruction. Accordingly, Hasegawa also fails to discloses or suggests at least the feature of "executing a branch instruction in execution of an instruction stream with the branch instruction including a first token that specifies the number of instructions in the instruction stream that are after the branch instruction to execute before performing the branch operation and a second token that specifies a branch guess operation to cause the processor, if the first token specifies zero or one instructions to execute after the branch, to prefetch, prior to performing an evaluation of a branch condition associated with the branch instruction, an instruction for the "branch taken" condition rather than a next sequential instruction," as required by applicant's independent claim 1.

Because neither Takayama nor Hasegawa discloses or suggests, alone or in combination, at least the feature of "executing a branch instruction in execution of an instruction stream with the branch instruction including a first token that specifies the number of instructions in the instruction stream that are after the branch instruction to execute before performing the branch operation and a second token that specifies a branch guess operation to cause the processor, if the first token specifies zero or one instructions to execute after the branch, to prefetch, prior to performing an evaluation of a branch condition associated with the branch instruction, an instruction for the "branch taken" condition rather than a next sequential instruction," applicant's independent claim 1 is patentable over the cited art.

Furthermore, applicant contends that there would also be no motivation to combine the teachings of Hasegawa with the teachings of Takayama. Particularly, Takayama's branch prediction scheme is implemented using a unique pipelined processor that enables determining which instruction to fetch after a branch prediction instruction is encountered, and dynamically updating subsequent branch prediction instructions. Specifically, Takayama explains:

> **FIG. 10 shows a flow of the pipeline process performed by the information**
> **processing apparatus 100. Here, the initial values of the branch prediction**

Applicant : Gilbert Wolrich et al.
Serial No. : 10/070,008
Filed : July 3, 2002
Page : 10 of 12

Attorney's Docket No.: 10559-311US1 / P9632US
Intel Corporation

information 20b and the branch history information 20c of the branch
instruction 20 stored in the main memory 9 are respectively "0" and "01".
It should be noted here that these initial values are generated by a compiler
through a static analysis on a source program. Also, a cache block including
the branch instruction is read from the main memory 9 and is stored in the
cache memory 11a.

(1) Instruction Fetching (IF) Stage

The branch instruction is read out from the cache memory 11a and is
stored in the instruction buffer 12a by the instruction fetching unit 12
during the second half of the clock cycle 1 (1b).

(2) Decoding (DEC) Stage

The branch instruction pre-read by the instruction buffer 12a is transferred
to the instruction register 13 and is decoded by the instruction decoding
unit 14 during the first half of the clock cycle 2 (2a).

More specifically, the instruction decoding unit 14 judges whether the
instruction to be decoded is a branch instruction by decoding the operation
code 13a stored in the instruction register 13. Also, in addition to outputting
the control signal for the judgement of the branch condition to the
instruction executing unit 15, the instruction decoding unit 14 informs the
address control unit 8 of the branch prediction information 13b stored in
the instruction register 13.

The address control unit 8 predicts that the execution of the instruction will
not result in a branch because the branch prediction information 13b is
"0", and then increments the value of the instruction fetch counter 10a by
controlling the address generating unit 19. The value (the fetch address) is
then outputted to the cache unit 11 through the instruction fetching unit 12.

As a result, in the IF stage during the second half of the clock cycle 2 (2b),
the next instruction following the present branch instruction is read out by
the instruction fetching unit 12 and is stored in the instruction buffer 12a.
Accordingly, the next instruction is pre-read in accordance with the branch
prediction information included in the branch instruction even before the
branch instruction is executed.

(3) Executing (EX) Stage

During the first half of the clock cycle 3 (3a), the instruction executing unit
15 judges the condition of the branch instruction in accordance with the
control signal from the instruction decoding unit 14. After this, the
instruction executing unit 15 informs the branch history information
generating unit 16 and the branch prediction information generating unit
17 of the execution result (whether the branch was taken or not taken), as
well as informing the instruction decoding unit 14 and the address control
unit 8 of the prediction result (whether the prediction was correct or
incorrect).

On receiving the notification of the execution result during the second half
of the clock cycle 3 (3b), the branch history information generating unit 16
and the branch prediction information generating unit 17 obtain the branch
history information 13c stored in the instruction register 13, generate new

Applicant : Gilbert Wolrich et al.
Serial No. : 10/070,008
Filed : July 3, 2002
Page : 11 of 12

Attorney's Docket No.: 10559-311US1 / P9632US
Intel Corporation

> branch history information and new branch prediction information or
> maintain the present information, and then inform the branch instruction
> updating unit 18 of the result. Having received the notification of the new
> branch history information and/or the new branch prediction information,
> the branch instruction updating unit 18 generates the first 16 bits of the
> new branch instruction including these sets of information.
>
> It should be noted here that when the prediction is incorrect, the penalty
> processing (invalidation of the instruction transferred by the instruction
> register 13 and fetching the next instruction to be executed based on the
> execution result) is performed by the instruction executing unit 15 and the
> instruction fetching unit 12. (emphasis added, col. 12, line 53, to col. 13,
> line 58

Thus, the timing requirements of Takayama's pipeline processing structure, particularly
the timing requirement of the Decoding (DEC) stage and the Execution (EX) stage, preclude the
possibility of using any type of branch delay functionality in conjunction with Takayama's
branch prediction scheme and implementation. Accordingly, to combine Hasegawa's branch
delay scheme and implementation with Takayama's branch prediction scheme and
implementation would change the principle of operation of Takayama's branch prediction
implementation and/or render Takayama completely unsatisfactory for its intended purpose and
is therefore not suggested.

For the foregoing reasons, applicant contends that independent claim 1 is patentable over
the cited art.

Claims 3-4, 6-8, 10-13 and 24 depend from independent claim 1 and are therefore
patentable for at least the same reasons as independent claim 1.

Independent claims 17 and 22 recite "evaluating a second token that specifies a branch
guess operation, which causes the processor, if the first token specifies zero or one instructions to
execute after the branch, to prefetch, prior to performing an evaluation of a branch condition
associated with the branch instruction, an instruction for the "branch taken" condition rather than
a next sequential instruction," or similar language. For reasons similar to those provided with
respect to independent claim 1, at least these features are patentable over the cited art.

Claims 19-21 and 25 depend from independent claim 17 and are therefore patentable for
at least the same reasons as independent claim 17.

Claim 26 depends from applicant's independent claim 22 and is therefore patentable for
at least the same reasons as independent claim 22.

Applicant : Gilbert Wolrich et al.
Serial No. : 10/070,008
Filed : July 3, 2002
Page : 12 of 12

Attorney's Docket No.: 10559-311US1 / P9632US
Intel Corporation

It is believed that all the rejections and/or objections raised by the examiner have been addressed.

In view of the foregoing, applicant respectfully submits that the application is in condition for allowance and such action is respectfully requested at the examiner's earliest convenience.

All of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable.

Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

Enclosed is a Request for Continued Examination and a Petition for Two Month Extension of Time. The fees in the amount of $790 and $450 are being paid concurrently on the Electronic Filing System (EFS) by way of Deposit Account authorization. Please apply any other required fees to deposit account 06-1050, referencing the attorney docket number shown above.

Respectfully submitted,

Date: _March 9, 2007_

Ido Rabinovitch
Attorney for Intel Corporation
Reg. No. L0080

PTO Customer No. 20985
Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

21578344.doc